Git Assignment

Q 1. a. Create new folder

Ans : Mkdir git_folder

Cd git_folder

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell (master)
$ mkdir git_folder

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell (master)
$ cd git_folder
```

b. b. Put the following file in the folder

.code.txt

.log.txt

.output.txt

Using touch command we can make this file

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ touch code.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ touch log.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ touch output.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ ls
code.txt  log.txt  output.txt
```

C. Stage the code.txt and output.txt file

For staging we will use git add file_name command

step 1: Git add code.txt

step 2: Git add output.txt

Step 3: Git status

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git init
Initialized empty Git repository in C:/Users/lalba/shell/git_folder/.git/

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        code.txt
        log.txt
        output.txt

nothing added to commit but untracked files present (use "git add" to track)

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git add code.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git add output.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code.txt
        new file:   output.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        log.txt


lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$
```

d. commit them

Using the below command we can commit the file

git commit -m “add comment”

E. And finally push them to git

Add the origin using below command to push the file

git remote add origin https://github.com/Lal44/git_folder.git

push the file using below command

git push origin master

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git commit -m "commit the code.txt and output.txt file"
[master (root-commit) fd65dff] commit the code.txt and output.txt file
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 code.txt
 create mode 100644 output.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git push main origin
error: src refspec origin does not match any
error: failed to push some refs to 'main'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git remote add origin https://github.com/Lal44/git_folder.git

lalba@LAPTOP-PHMV5VKC MINGW64 ~/shell/git_folder (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lal44/git_folder.git
 * [new branch]      master -> master
```

2. . Please share the commands for the above point

Step1: git init

Step2: git add file_name

Step3: git commit -m “add commit”

Step4: git push origin master

Q2. Task to Be Performed:

1. Create a Git working directory with feature1.txt and feature2.txt in the master branch

   We have created two working directory feature1 and feature2

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~ (master)
$ mkdir myrepo

lalba@LAPTOP-PHMV5VKC MINGW64 ~ (master)
$ cd myrepo

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ ls

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git init
Initialized empty Git repository in C:/Users/lalba/myrepo/.git/

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ touch feature{1..2}.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ ls
feature1.txt  feature2.txt
```

2. Create 3 branches develop, feature1 and feature

   Ans:
   Step1 : git checkout master
   Step2 : git checkout -b devops
   Step3 : git checkout -b feature1
   Step4 : git checkout -b feature2

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git branch
  devops
  feature1
  feature2
* master

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$
```

Q 3.In develop branch create develop.txt, do not stage or commit it

Step 1: Git checkout -b devops

Step 2: touch devops.txt

Step 3: git add devops.txt

Step 4: git stash

Sept 5: git stash list(to verify the file is stash or not)

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ touch devops.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ ls
devops.txt  feature1.txt  feature2.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git add devops.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git status
On branch devops
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   devops.txt


lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git stash
Saved working directory and index state WIP on devops: 567d623 add feature1 and featur2

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git stash lish
fatal: subcommand wasn't specified; 'push' can't be assumed due to unexpected token 'lish'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git stash list
stash@{0}: WIP on devops: 567d623 add feature1 and featur2

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$
```

Q 4.Stash this file and check out to feature1 branch

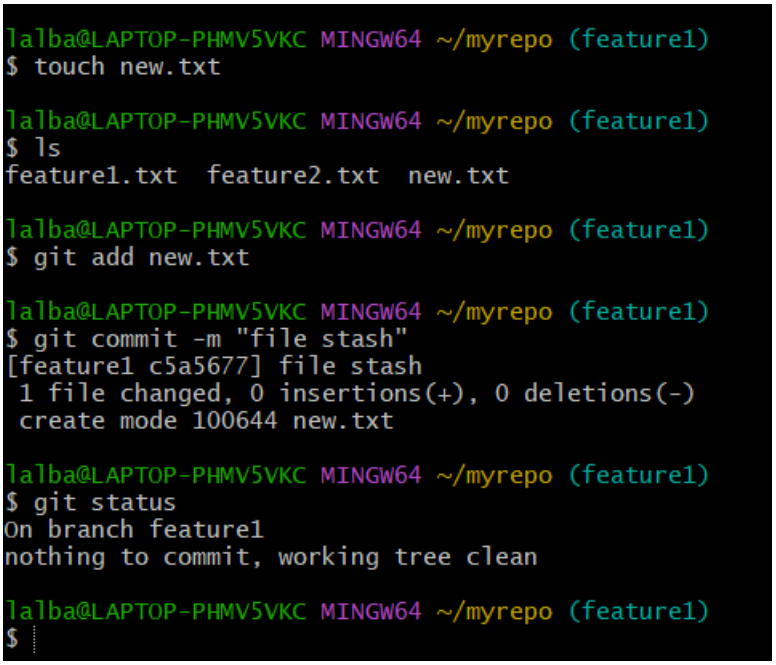
Ans : step 1: git stash
Step 2: git checkout feature1

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git checkout feature1
Switched to branch 'feature1'
```

Q 5.Create new.txt file in feature1 branch, stage and commit

This file

Ans : step 1: git checkout feature1
Step 2: touch new.txt
Step 3: git add new.txt
Step 4: git commit -m “file new.txt stage”

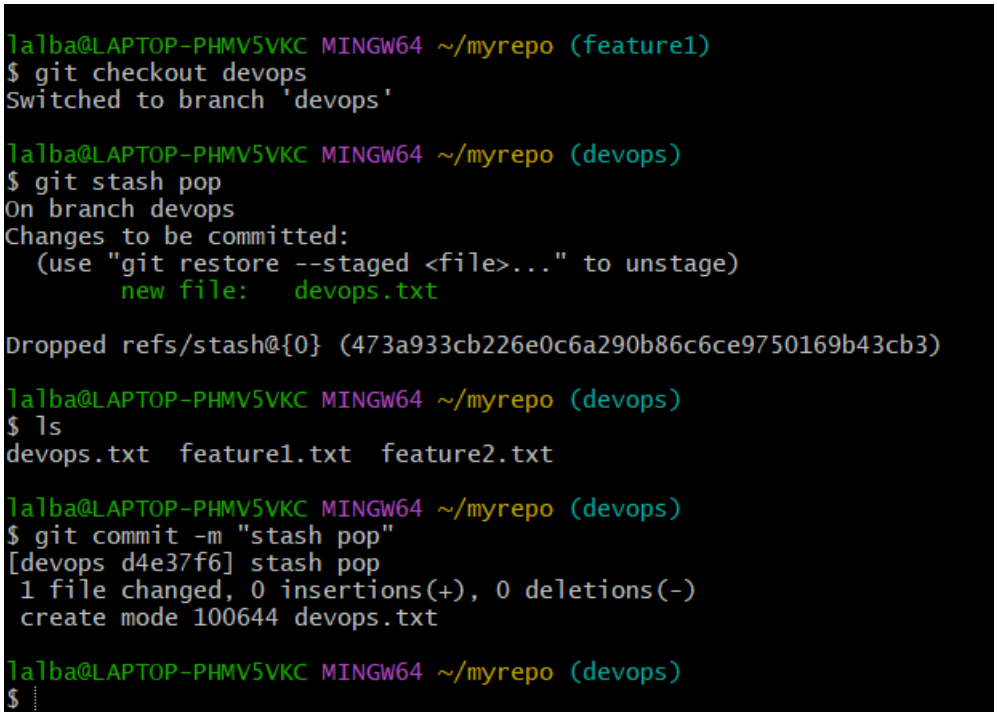


Q 6: Checkout the develop, unstash this file and commit

Ans : Step 1: git checkout devops

Step 2: git stash pop

Step 3: git commit -m “comment”



Q 7. Please submit all the Git commands used to do the above step

Ans:

1. Create a Git working directory and commit files to master branch

mkdir myrepo

cd myrepo

git init

touch feature1.txt feature2.txt

echo "Content for feature1" > feature1.txt

echo "Content for feature2" > feature2.txt

git add feature1.txt feature2.txt

git commit -m "Add feature1.txt and feature2.txt to the master branch"

2. Create branches

git checkout -b develop

git checkout master

git checkout -b feature1

git checkout master

git checkout -b feature2

3. In develop branch, create develop.txt (do not stage or commit)

git checkout develop

touch develop.txt

Do not stage or commit

4. Stash the file and switch to feature1 branch

git stash push -m "Stash develop.txt"

git checkout feature1

5. Create, stage, and commit new.txt in feature1 branch

touch new.txt

git add new.txt

git commit -m "Add new.txt in feature1 branch"

6. Checkout to develop, unstash develop.txt, and commit

git checkout develop

git stash pop

git add develop.txt

git commit -m "Add develop.txt to develop branch"

Q 3. Task to Be Performed:

1. Create a Git working directory, with the following branches:

Develop

F1

f2

Ans :

Step 1: mkdir myrepo

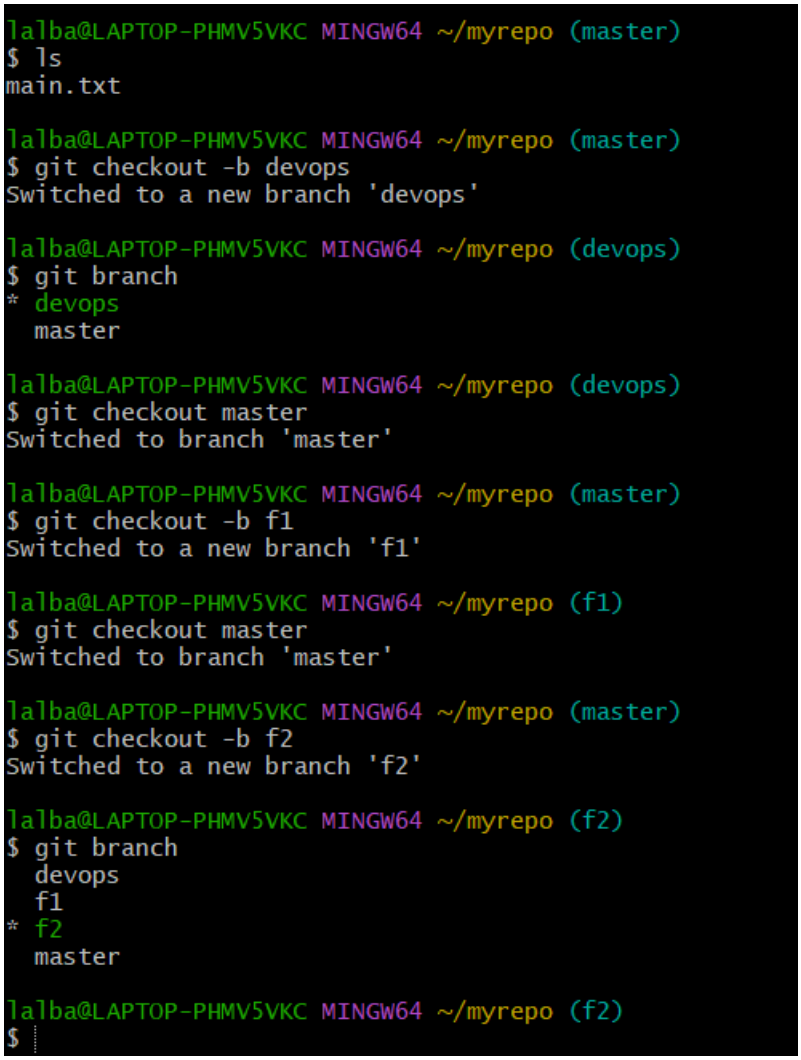
Step2: git inti

Step3: git checkout -b devops

Step4: git checkout -b f1

Step5: git checkout -b f2

Step6: git status



3. In the master branch, commit main.txt file

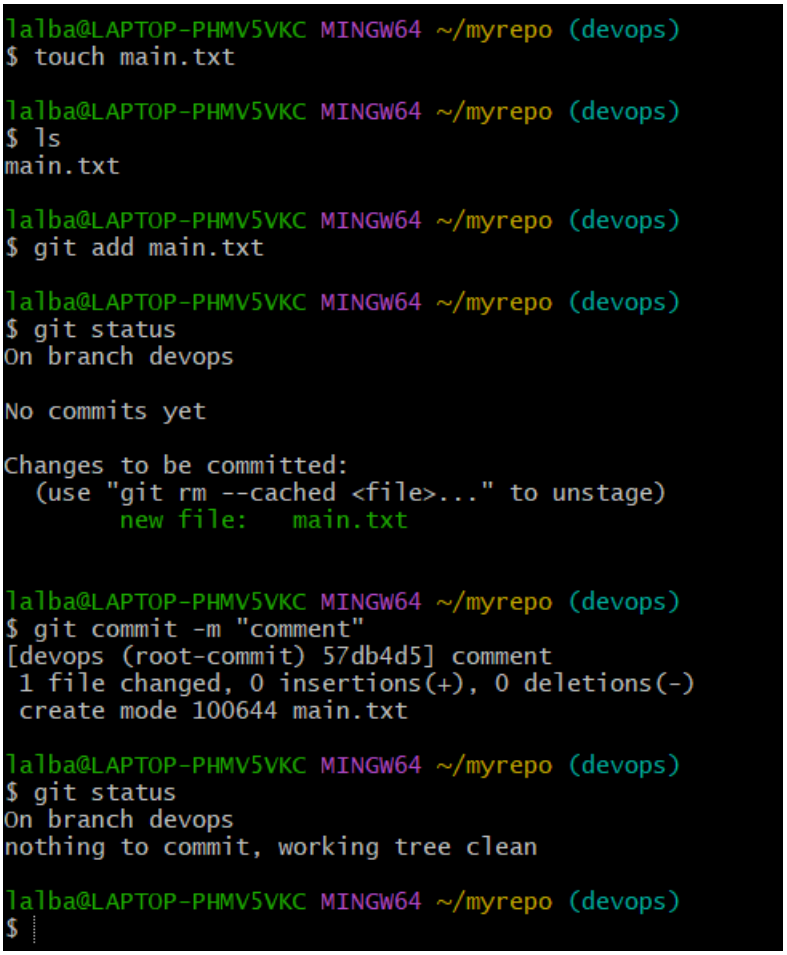
   Ans:

   Step 1:touch main.txt

   Step 2: git add main.txt

   Step 3: git comment -m “comment”

   Step 4: git status



4. Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively

   Ans:
   Step1: git checkout devops
   Step2: touch devops.txt
   Step3: git checkout f1
   Step4: touch f1.txt
   Step5: git checkout f2
   Step6: touch f2.txt

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f2)
$ git checkout devops
Switched to branch 'devops'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ touch devops.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ ls
devops.txt  main.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (devops)
$ git checkout f1
Switched to branch 'f1'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f1)
$ touch f1.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f1)
$ ls
devops.txt  f1.txt  main.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f1)
$ git checkout f2
Switched to branch 'f2'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f2)
$ touch f2.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f2)
$ ls
devops.txt  f1.txt  f2.txt  main.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (f2)
$
```

5. Push all these branches to GitHub

Ans :

```
# Add GitHub repository as remote
git remote add origin <your-repo-url>
# Push master branch
git push -u origin master
# Push develop branch
git push -u origin develop
# Push F1 branch
git push -u origin F1
# Push f2 branch
git push -u origin f2
```

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 205 bytes | 205.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lal44/myrepo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git push -u origin devops
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 237 bytes | 237.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'devops' on GitHub by visiting:
remote:      https://github.com/Lal44/myrepo/pull/new/devops
remote:
To https://github.com/Lal44/myrepo.git
 * [new branch]      devops -> devops
branch 'devops' set up to track 'origin/devops'.

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git push -u origin f1
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 228 bytes | 228.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'f1' on GitHub by visiting:
remote:      https://github.com/Lal44/myrepo/pull/new/f1
remote:
To https://github.com/Lal44/myrepo.git
 * [new branch]      f1 -> f1
branch 'f1' set up to track 'origin/f1'.

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git push -u origin f2
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 229.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'f2' on GitHub by visiting:
remote:      https://github.com/Lal44/myrepo/pull/new/f2
remote:
To https://github.com/Lal44/myrepo.git
 * [new branch]      f2 -> f2
branch 'f2' set up to track 'origin/f2'.
```

6. On local delete f2 branch

   Ans :

git branch -d f2

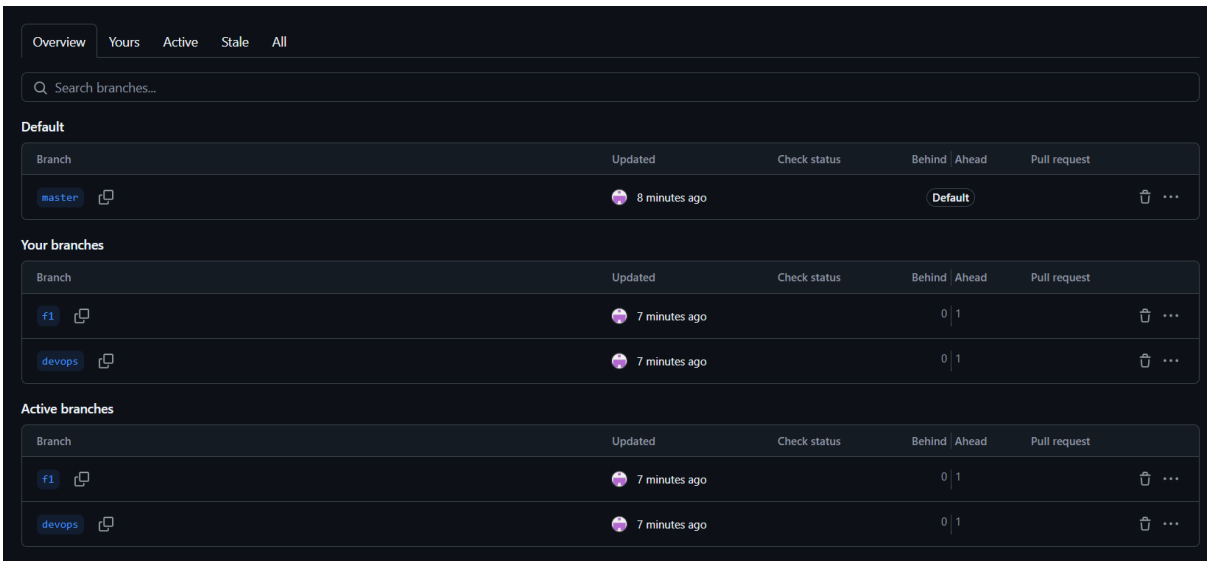
```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git branch -d f2
warning: deleting branch 'f2' that has been merged to
         'refs/remotes/origin/f2', but not yet merged to HEAD
Deleted branch f2 (was 18e962e).

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git branch
  devops
  f1
* master
```

7. Delete the same branch on GitHub as well

   Ans:

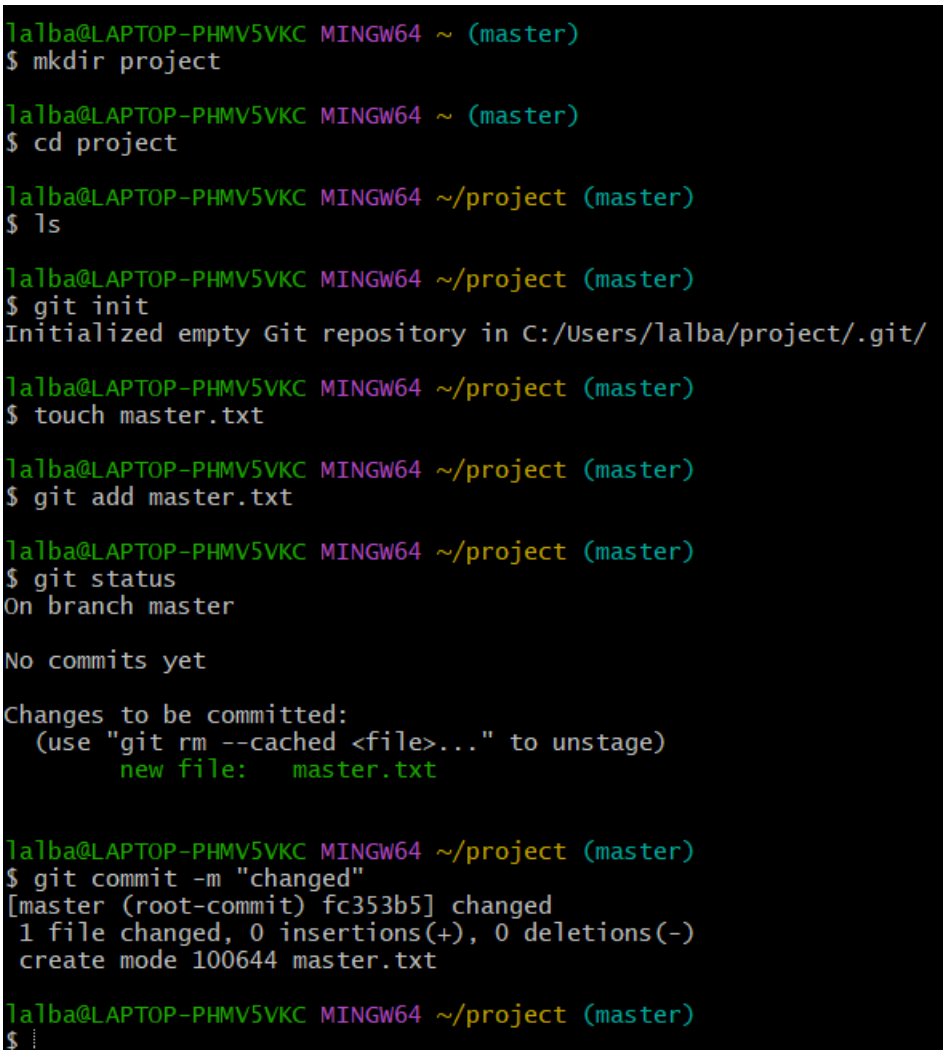
   Git push origin –delete f2

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git push origin --delete f2
To https://github.com/Lal44/myrepo.git
 - [deleted]         f2

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$ git branch
  devops
  f1
* master

lalba@LAPTOP-PHMV5VKC MINGW64 ~/myrepo (master)
$
```



4 Task to Be Performed:

1. Put master.txt on master branch, stage and commit
   Ans:
   Step1: mkdir project
   Step2: git init
   Step3: touch master.txt
   Step4:git add master.txt
   Step5: git commit -m “comment”



2. Create 3 branches: public 1, public 2 and private

Ans:

```
git checkout -b public1

# Switch back to master
git checkout master

# Create and switch to the public2 branch
git checkout -b public2

# Switch back to master
git checkout master

# Create and switch to the private branch
git checkout -b private
```

3. Put public1.txt on public 1 branch, stage and commit

Ans:

```
# Switch to the public1 branch
git checkout public1

# Create public1.txt file
touch public1.txt
echo "Content for public1.txt" > public1.txt

# Stage and commit public1.txt
git add public1.txt
```

git commit -m "Add public1.txt to public1 branch"

4. Merge public 1 on master branch
   Ans:
   # Switch to master branch
   git checkout master

   # Merge public1 branch into master
   git merge public1

5. Merge public 2 on master branch

   Ans:
   # Switch to public2 branch
   git checkout public2

   # Merge public2 branch into master
   git checkout master
   git merge public2

6. Edit master.txt on private branch, stage and commit
   Ans:
   # Switch to private branch
   git checkout private

   # Edit master.txt file
   echo "Updated content on private branch" >
   master.txt

Stage and commit the change
git add master.txt
git commit -m "Update master.txt on private branch"

7. Now update branch public 1 and public 2 with new master code in private
   Ans:
   # Switch to public1 branch
   git checkout public1

   # Merge master branch (which includes updates from private) into public1
   git merge master

   # Switch to public2 branch
   git checkout public2

   # Merge master branch (which includes updates from private) into public2
   git merge master

8. Also update new master code on master
   Ans:
   # Switch to master branch
   git checkout master

   # Merge private branch into master
   git merge private

9. Finally update all the code on the private branch
   Ans:
   # Switch to private branch
   git checkout private

   # Merge master branch into private to ensure it has the latest changes
   git merge master

5 Tasks to Be Performed:

1. Create a Git Flow workflow architecture on Git
   Ans:

   Git Flow is a branching model that defines a set of branches for managing development. Here's the architecture:

   master: Production-ready branch. Code here should always be stable and deployable.
   develop: Integration branch where features are merged before a release.
   feature/*: Branches for developing new features, created off of develop.
   release/*: Branches for preparing new releases, created off of develop.
   hotfix/*: Branches for urgent fixes, created off of master.

2. Create all the required branches

   Ans:
   git flow init

   Create the develop branch if it doesn’t exist:
   git checkout -b develop
   git push -u origin develop

   Create a feature/* branch:
   git flow feature start my-feature

   Create a release/* branch (optional, when preparing for a release):
   git flow release start 1.0.0

   Create a hotfix/* branch (optional, for urgent fixes):
   git flow hotfix start urgent-fix

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (main)
$ git branch
* main

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (main)
$ git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main] main
Branch name for "next release" development: [develop] development

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/lalba/lalbabu/gitflow-on-github/.git/hooks]

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ ls
README.md

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git push origin development
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'development' on GitHub by visiting:
remote:      https://github.com/Lal44/gitflow-on-github/pull/new/development
remote:
To https://github.com/Lal44/gitflow-on-github.git
 * [new branch]      development -> development

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git branch
* development
  main
```

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (feature/my-feature)
$ git branch
  development
* feature/my-feature
  main

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (feature/my-feature)
$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'development'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '1.0.0'


lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git flow hotfix start urgent-fix
Switched to a new branch 'hotfix/urgent-fix'

Summary of actions:
- A new branch 'hotfix/urgent-fix' was created, based on 'main'
- You are now on branch 'hotfix/urgent-fix'

Follow-up actions:
- Start committing your hot fixes
- Bump the version number now!
- When done, run:

     git flow hotfix finish 'urgent-fix'


lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$ git branch -a
  development
  feature/my-feature
* hotfix/urgent-fix
  main
  release/1.0.0
  remotes/origin/HEAD -> origin/main
  remotes/origin/development
  remotes/origin/main

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$
```

4.starting from feature branch, push the branch to the master, following the architecture

Ans:

Step1: git checkout -b feature/my-feature

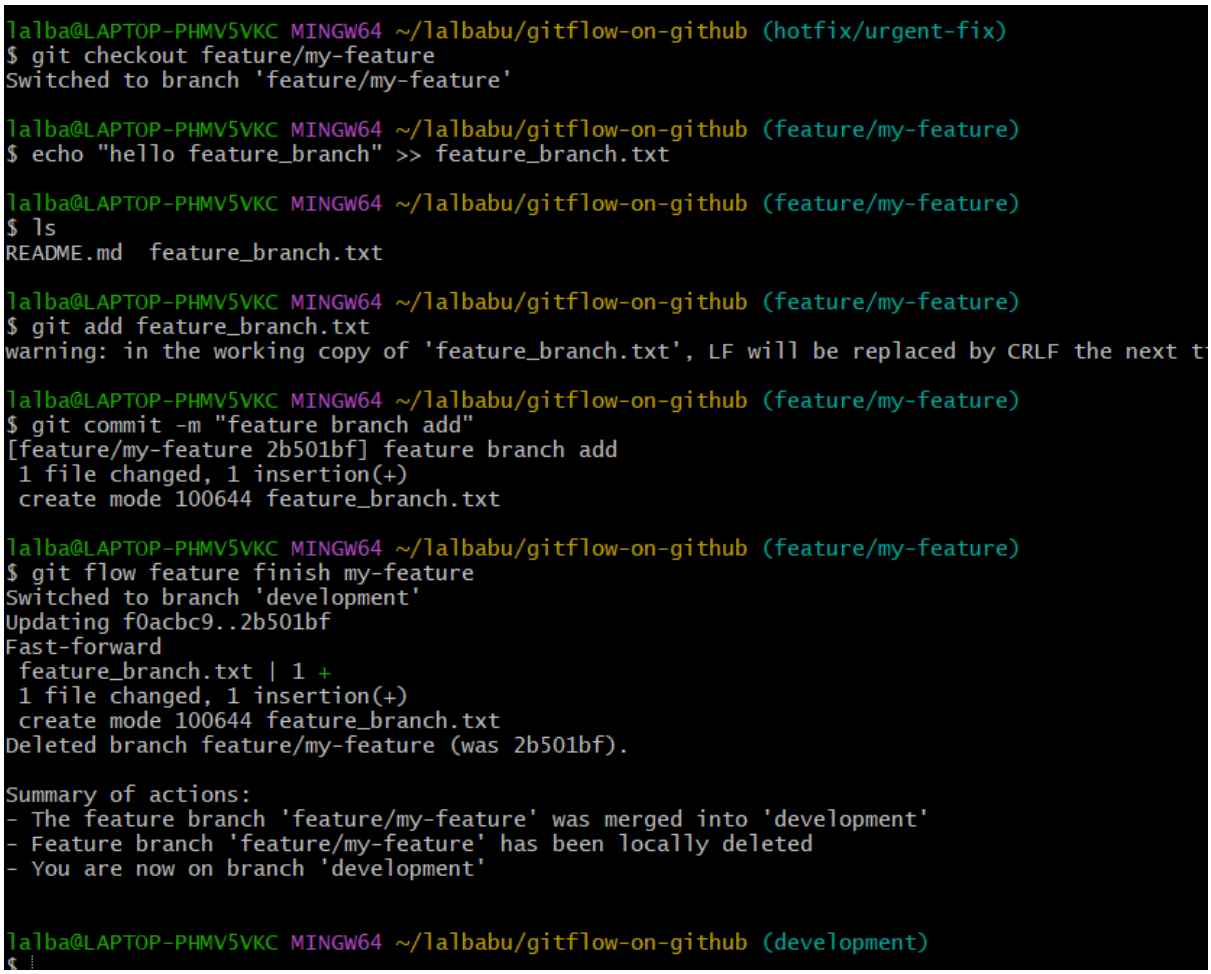
Step2: echo “hello feature_branch” >> feature.txt

Step3: git add feature.txt

Step4: git commit -m “feature-branch modified”

Step4:once task complete in feature branch we will move to development branch using below command

git flow feature finish my-feature



After that we will push the above change in feature-branch file to development branch using below command.

git push --set-upstream origin development or

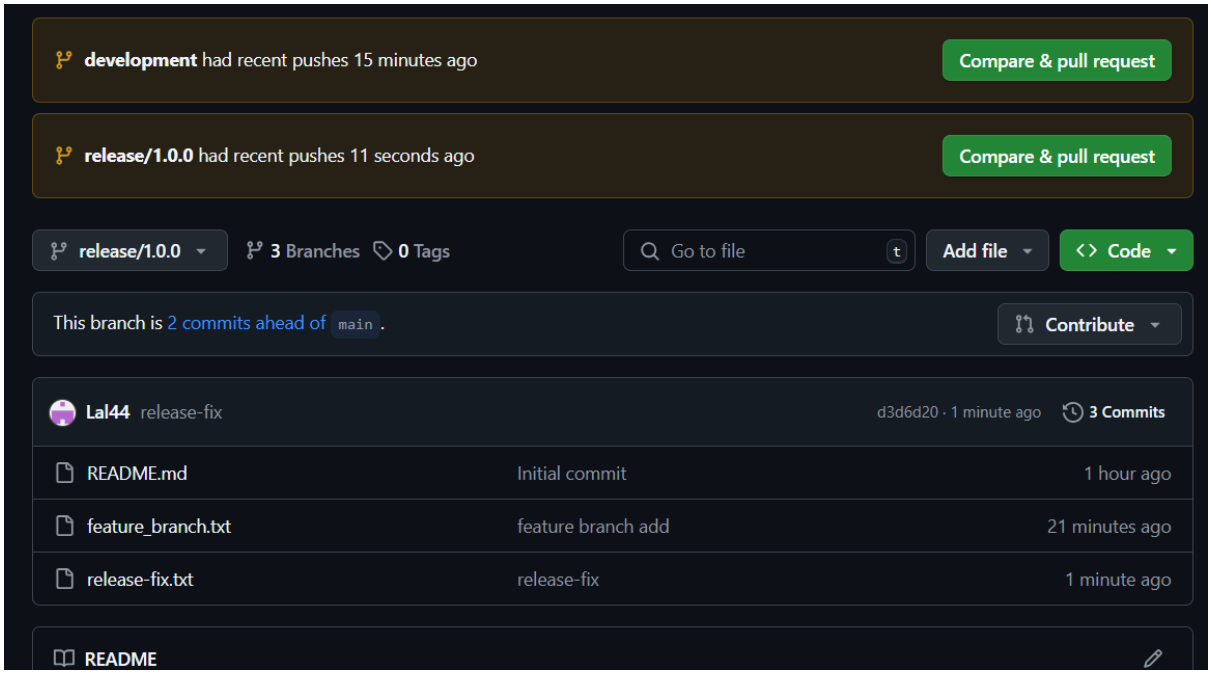
git push origin

Prepare for a Release: Create a release branch to prepare for merging into master:

git flow release start 1.0.0

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'development'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '1.0.0'


lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git branch
  development
  hotfix/urgent-fix
  main
* release/1.0.0

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git push origin
fatal: The current branch release/1.0.0 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin release/1.0.0

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git push --set-upstream origin release/1.0.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'release/1.0.0' on GitHub by visiting:
remote:      https://github.com/Lal44/gitflow-on-github/pull/new/release/1.0.0
remote:
To https://github.com/Lal44/gitflow-on-github.git
 * [new branch]      release/1.0.0 -> release/1.0.0
branch 'release/1.0.0' set up to track 'origin/release/1.0.0'.

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
```

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ ls
README.md  feature_branch.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ echo "this is fix in release branch" >> release-fix.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git add release-fix.txt
warning: in the working copy of 'release-fix.txt', LF will be replaced by CRLF the next time Git touches it

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git commit -m "release-fix"
[release/1.0.0 d3d6d20] release-fix
 1 file changed, 1 insertion(+)
 create mode 100644 release-fix.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$ git push origin
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 347 bytes | 347.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lal44/gitflow-on-github.git
   2b501bf..d3d6d20  release/1.0.0 -> release/1.0.0

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (release/1.0.0)
$
```



Finish the Release Branch: Complete the release and merge it into master and development:

git flow release finish '1.0.0'

4.Push an urgent.txt on master using hotfix

Start a hotfix branch:

git flow hotfix start urgent-fix

Add the urgent.txt file:

echo "Urgent fix content" > urgent.txt

git add urgent.txt

git commit -m "Add urgent.txt for hotfix"

Finish the hotfix branch:

git flow hotfix finish 'urgent-fix'

This merges the hotfix changes into master and develop.

Push changes to master:

git push origin master

git push origin develop

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (main)
$ git checkout hotfix/urgent-fix
Switched to branch 'hotfix/urgent-fix'

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$ echo "Urgent fix content" > urgent.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$ git add urgent.txt
warning: in the working copy of 'urgent.txt', LF will be replaced by CRLF the next time Git touches it

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$ git commit -m "Add urgent.txt for hotfix"
[hotfix/urgent-fix 38953c0] Add urgent.txt for hotfix
 1 file changed, 1 insertion(+)
 create mode 100644 urgent.txt

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (hotfix/urgent-fix)
$ git flow hotfix finish 'urgent-fix'
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Merge made by the 'ort' strategy.
 urgent.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 urgent.txt
Switched to branch 'development'
Your branch is up to date with 'origin/development'.
Merge made by the 'ort' strategy.
 release-fix.txt | 1 +
 urgent.txt      | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 release-fix.txt
 create mode 100644 urgent.txt
Deleted branch hotfix/urgent-fix (was 38953c0).

Summary of actions:
- Hotfix branch 'hotfix/urgent-fix' has been merged into 'main'
- The hotfix was tagged 'urgent-fix'
- Hotfix tag 'urgent-fix' has been back-merged into 'development'
- Hotfix branch 'hotfix/urgent-fix' has been locally deleted
- You are now on branch 'development'
```

```
lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git push origin
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 778 bytes | 778.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Lal44/gitflow-on-github.git
   2b501bf..d98478d  development -> development

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lal44/gitflow-on-github.git
   1111716..dc7fab8  main -> main

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
$ git push origin development
Everything up-to-date

lalba@LAPTOP-PHMV5VKC MINGW64 ~/lalbabu/gitflow-on-github (development)
```